AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A method for executing a fail instruction to facilitate transactional execution on a processor, comprising:

transactionally executing a block of instructions within a program;

wherein changes made during the transactional execution are not committed to the architectural state of the processor unless the transactional execution successfully completes; and

~~if the fail instruction is encountered during the transactional execution, terminating the transactional execution without committing results of the transactional execution to the architectural state of the processor, wherein terminating the transactional execution involves branching to a location specified by the fail instruction or to a location specified by a start transactional execution (STE) instruction at the beginning of the transactional execution; or setting state information in the processor and continuing the transactional execution, wherein the processor handles the failure later.~~

if the fail instruction is encountered during the transactional execution, setting state information within the processor to indicate that the fail instruction was encountered and continuing transactional execution; and

performing one or more failure actions based on the state information at a later time.

1   2. (Currently Amended) The method of claim 1, <u>wherein performing one</u>

2   <u>or more failure actions</u> ~~wherein terminating the transactional execution~~ involves

3   <u>terminating the transactional execution and</u> discarding changes made during the

4   transactional execution.

1   3. (Original) The method of claim 2, wherein discarding changes made

2   during the transactional execution involves:

3   discarding register file changes made during the transactional execution;

4   clearing load marks from cache lines;

5   draining store buffer entries generated during transactional execution; and

6   clearing store marks from cache lines.

1   4-5. (Cancelled)

1   6. (Currently Amended) The method of ~~claim 1~~<u>claim 2</u>, wherein

2   terminating the transactional execution additionally involves attempting to re-

3   execute the block of instructions.

1   7. (Previously Presented) The method of claim 1, wherein if the

2   transactional execution of the block of instructions is successfully completed, the

3   method further comprises:

4   atomically committing changes made during the transactional execution;

5   and

6   resuming normal non-transactional execution.

1   8. (Original) The method of claim 1, wherein potentially interfering data

2   accesses from other processes are allowed to proceed during the transactional

3   execution of the block of instructions.

1   9. (Original) The method of claim 1, wherein if an interfering data access

2 from another process is encountered during the transactional execution, the

3 method further comprises:

4   discarding changes made during the transactional execution; and

5   attempting to re-execute the block of instructions.

1   10. (Original) The method of claim 1, wherein the block of instructions to

2 be executed transactionally comprises a critical section.

1   11. (Original) The method of claim 1, wherein the fail instruction is a

2 native machine code instruction of the processor.

1   12. (Original) The method of claim 1, wherein the fail instruction is

2 defined in a platform-independent programming language.

1   13. (Currently Amended) A computer system that supports a fail

2 instruction to facilitate transactional execution, comprising:

3   a processor; and

4   an execution mechanism within the processor;

5   wherein the execution mechanism is configured to transactionally execute

6 a block of instructions within a program;

7   wherein changes made during the transactional execution are not

8 committed to the architectural state of the processor unless the transactional

9 execution successfully completes; and

10   wherein if the fail instruction is encountered during the transactional

11 execution, the execution mechanism is configured to:

12   ~~terminate the transactional execution without committing results of~~

13   ~~the transactional execution to the architectural state of the processor,~~

4

14   ~~wherein terminating the transactional execution involves branching to a~~

15   ~~location specified by the fail instruction or to a location specified by a start~~

16   ~~transactional execution (STE) instruction at the beginning of the~~

17   ~~transactional execution, or to set state information in the processor and~~

18   ~~continue transactional execution, wherein the execution mechanism is~~

19   ~~configured to handle the failure later~~

20   <u>set state information within the processor to indicate that the fail</u>

21 <u>instruction was encountered and continue transactional execution; and</u>

22   <u>perform one or more failure actions based on the state information at a</u>

23 <u>later time</u>.

1   14. (Currently Amended) The computer system of claim 13, wherein while

2 ~~terminating the transactional execution~~<u>performing the one or more failure actions</u>,

3 the execution mechanism is configured to <u>terminate the transactional execution</u>

4 <u>and</u> discard changes made during the transactional execution.

1   15. (Original) The computer system of claim 14, wherein while discarding

2 changes made during the transactional execution, the execution mechanism is

3 configured to:

4   discard register file changes made during the transactional execution;

5   clear load marks from cache lines;

6   drain store buffer entries generated during transactional execution; and to

7   clear store marks from cache lines.

1   16-17. (Cancelled)

1   18. (Currently Amended) The computer system of ~~claim 13~~claim 14,
2   wherein while terminating the transactional execution, the execution mechanism
3   is additionally configured to attempt to re-execute the block of instructions.

1   19. (Previously Presented) The computer system of claim 13, wherein if
2   the transactional execution of the block of instructions is successfully completed,
3   the execution mechanism is configured to:
4   atomically commit changes made during the transactional execution; and
5   to
6   resume normal non-transactional execution.

1   20. (Original) The computer system of claim 13, wherein the computer
2   system is configured to allow potentially interfering data accesses from other
3   processes to proceed during the transactional execution of the block of
4   instructions.

1   21. (Original) The computer system of claim 13, wherein if an interfering
2   data access from another process is encountered during the transactional
3   execution, the execution mechanism is configured to:
4   discard changes made during the transactional execution; and to
5   attempt to re-execute the block of instructions.

1   22. (Original) The computer system of claim 13, wherein the block of
2   instructions to be executed transactionally comprises a critical section.

1   23. (Original) The computer system of claim 13, wherein the fail
2   instruction is a native machine code instruction of the processor.

1   24. (Original) The computer system of claim 13, wherein the fail

2   instruction is defined in a platform-independent programming language.

1   25. (Currently Amended) A computing means that supports a fail

2   instruction to facilitate transactional execution, comprising:

3   a processing means; and

4   an execution means within the processing means;

5   wherein the execution means is configured to transactionally execute a

6   block of instructions within a program;

7   wherein changes made during the transactional execution are not

8   committed to the architectural state of the processor unless the transactional

9   execution successfully completes; and

10   wherein if the fail instruction is encountered during the transactional

11   execution, the execution means is configured to:

12   ~~terminate the transactional execution without committing results of~~

13   ~~the transactional execution to the architectural state of the processor,~~

14   ~~wherein terminating the transactional execution involves branching to a~~

15   ~~location specified by the fail instruction or to a location specified by a start~~

16   ~~transactional execution (STE) instruction at the beginning of the~~

17   ~~transactional execution, or to set state information in the processor and~~

18   ~~continue transactional execution, wherein the execution means is~~

19   ~~configured to handle the failure later~~

20   set state information within the processor to indicate that the fail

21   instruction was encountered and continue transactional execution; and

22   perform one or more failure actions based on the state information at a

23   later time.

1   26.   (New) The method of claim 1, wherein performing one or more

2   failure actions based on the state information at the later time involves performing

3   the failure actions when a commit instruction is encountered at an end of the

4   transaction.

1   27.   (New) The computer system of claim 13, wherein when

2   performing the one or more failure actions at the later time, the execution

3   mechanism is configured to perform the failure actions upon encountering a

4   commit instruction at an end of the transaction.

1   28.   (New) The computing means of claim 25, wherein when

2   performing the one or more failure actions at the later time, the execution means

3   is configured to perform the failure actions upon encountering a commit

4   instruction at an end of the transaction.